UE Informatique

Philippe EZEQUEL
Marc BERNARD
Émilie MORVANT
Baptiste JEUDY

L1 ST
2013-2014
Université Jean Monnet

Modalités pratiques

Planning

- début des TD la semaine prochaine ;
- 2 séances de TD par semaine paire ;
- 1 séance de TD par semaine impaire ;
- la semaine prochaine est une semaine **paire**.

Contrôle des connaissances

- un contrôle début novembre (coefficient $\frac{1}{2}$)
- un contrôle à la fin du semestre (coefficient $\frac{1}{2}$)

Adresses

`http://webperso.univ-st-etienne.fr/~ezequel/1info/`

`ezequel@univ-st-etienne.fr`

Objectifs pédagogiques

- de l'UE entière : panorama partiel, partial et subjectif de l'Informatique, en tant que science ;
- 4 séquences (plus ou moins) indépendantes ;
- de ma séquence : algorithmes et programmes : existence, conception, faisabilité.

Avertissement

Ces transparents sont (volontairement ?) incomplets.

Plan du cours

1. Introduction
2. Calculabilité
3. Conception d'algorithmes (et de programmes)
4. Faisabilité pratique

Introduction : au début sont les programmes. . .

Algorithmes et programmes : algorithmes

Algorithme

Suite finie d'instructions permettant de résoudre un «problème»

Exemples

1. $(uv)' = u'v + uv'$
2. partition
3. mode d'emploi
4. recette de cuisine
5. . . .

Remarque

Destiné à un lecteur humain, éventuellement vague

Algorithmes et programmes : programmes

Programme

Algorithme écrit dans un langage compris par une machine

Exemples

1. fiches perforées de métier à tisser
2. Linux, Windows
3. ce document !
4. page WWW

Remarque

Destiné à une machine, *doit être correct et inambigu*

Un exemple de vrai programme

Extrait du noyau Linux

```
457f 464c 0101 0001 0000 0000 0000 0000
0002 0003 0001 0000 86a0 0804 0034 0000
7ae8 0000 0000 0000 0034 0020 0007 0028
0021 001e 0006 0000 0034 0000 8034 0804
8034 0804 00e0 0000 00e0 0000 0005 0000
0004 0000 0003 0000 0114 0000 8114 0804
8114 0804 0013 0000 0013 0000 0004 0000
0001 0000 0001 0000 0000 0000 8000 0804
8000 0804 61a5 0000 61a5 0000 0005 0000
```

C'est illisible : nécessité de langages «évolués», et donc de traducteurs . . .

Objectif : faire faire les calculs par une machine !

Astrolabe

- Calcul de la position des étoiles et des planètes.

Sage Ross, http ://en.wikipedia.org/wiki/Astrolabe

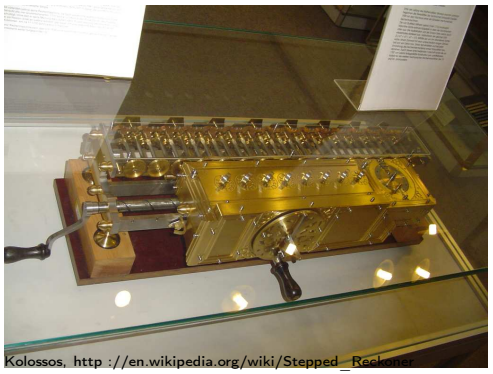
Pascaline

- Blaise Pascal, 1642, additions et soustractions.

D. Monniaud, http ://fr.wikipedia.org/wiki/Pascaline

Machine de Leibniz

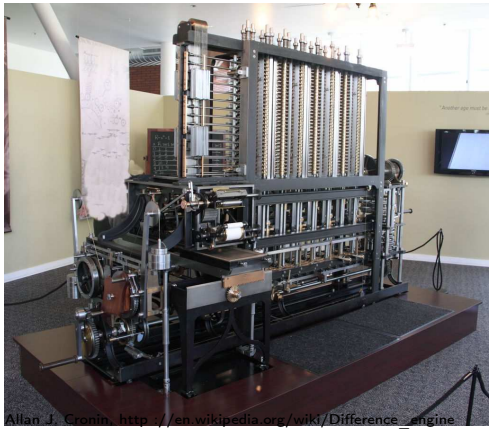
- 1694, 4 opérations (pas fabriquée à l'époque).



Kolossos, http ://en.wikipedia.org/wiki/Stepped_Reckoner

Machine à différences

- Charles Babbage, 1830, calcul de polynômes.



Allan J. Cronin, http ://en.wikipedia.org/wiki/Difference_engine

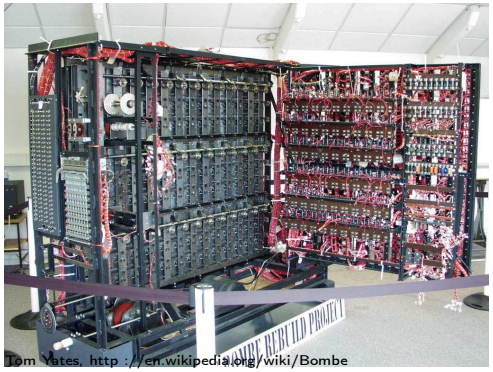
Tabulatrice

- Herman Hollerith, 1890.
- Comptage, 150 additions/s, machine électrique.
- Utilisée pour recensement USA 1890.

Stahlkocher, http ://en.wikipedia.org/wiki/Tabulating_machine

Bombe

- Utilisée pour déchiffrer les messages allemand (2e guerre mondiale)



Tom Yates, http ://en.wikipedia.org/wiki/Bombe

Le premier vrai ordinateur : Baby 1

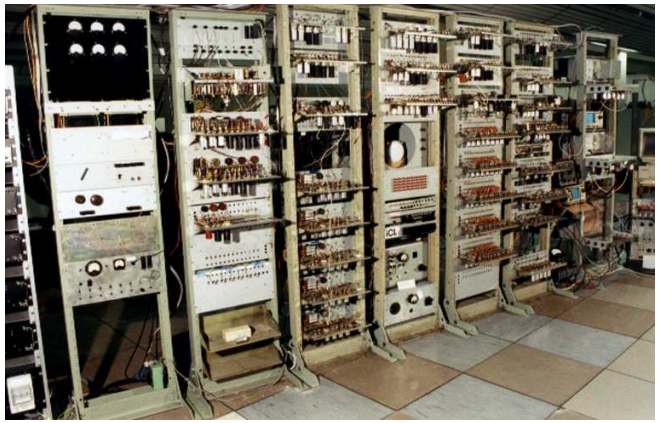
Quelques chiffres

- longueur 5,23 m, hauteur 3,26 m, poids 1 tonne ;
- voltage : de 350V à 1250V (lampes !) ;
- consommation 3500 W
- horloge à 100 kHz
- 500 instructions par seconde

Actuellement (juin 2013) : Tianhe-2 (Chine)

- consommation 17 MW
- 34.10^{15} instructions par seconde (34 **pétaflops**)

Une photo de Baby 1



Une photo de Tianhe-2

Modèles de calculabilité

Début du XX^{e} siècle : qu'est-ce que *calculer* ?

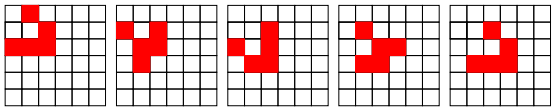
- système semi-Thue
- machine de Peano $\star$
- λ-calcul de Church
- algorithmes de Markov
- machine de Turing $\star$
- machine de von Neumann
- automate cellulaire de Conway $\star$
- clauses de Horn
- ...

Tous équivalents ! !

Automate cellulaire : le jeu de la vie

- damier infini ;
- chaque case (appelée **cellule**) est soit **vivante**, soit **morte** ;
- règles d'évolution :
  1. une cellule vivante entourée de 2 ou 3 cellules vivantes reste vivante, sinon elle meurt ;
  2. une cellule morte entourée de 3 cellules vivantes devient vivante, sinon elle reste morte.

Automate cellulaire : le jeu de la vie



Références :

1. une simulation en ligne : http ://www.bitstorm.org/gameoflife
2. la page Wikipedia «Jeu de la Vie»

machine de Peano

- évaluateur d'expressions arithmétiques
- possibilité de définir des fonctions

Expressions

(i) 0	0
(ii) $++$ Expr	eval(Expr)+1
(iii) $--$ Expr	eval(Expr)-1
(iv) si $E_1 = E_2$ alors E_3 sinon E_4	
(v) appel de fonction	

machine de Peano : exemples de définitions

Constante 2

```
DEUX = ++ ++ 0
```

Évaluation :

```
DEUX  =  ++ ++ 0
      =  (++ 0) + 1
      =  (0 + 1) + 1
      =  1 + 1
      =  2
```

Confort

On va supposer que la machine de Peano "connaît" tous les entiers...

machine de Peano

Expressions

(i) 0, 1, 2, 3, 4, ...
(ii) $++$ Expr
(iii) $--$ Expr
(iv) si $E_1 = E_2$ alors E_3 sinon E_4
(v) appel de fonction

machine de Peano : exemples de définitions

Addition : algorithme A

$X + 0 = X$

$X + Y = (X + (Y - 1)) + 1$

Addition : programme A

```
plus_A(X,Y) = si Y = 0 alors X sinon ++plus_A(X, --Y)
```


Programme A : exécution

Addition : programme A

```
plus_A(X,Y) = si Y = 0 alors X sinon ++plus_A(X, --Y)
```

```
plus_A(2,1)
        =  si 1 = 0 alors 2 sinon ++plus_A(2, --1)
        =   ++plus_A(2, --1)
        =   ++plus_A(2, 0)
        =   ++(si 0 = 0 alors 2 sinon ++plus_A(2, --0))
        =   ++ 2
        =   3
            CORRECT !
```


machine de Peano : exemples de définitions

Addition : algorithme B

$X + 0 = X$

$X + Y = (X + (Y+1)) - 1$

Addition : programme B

```
plus_B(X,Y) = si Y = 0 alors X sinon --plus_B(X, ++Y)
```


Programme 2 : exécution

Addition : programme B

```
plus_B(X,Y) = si Y = 0 alors X sinon --plus_B(X, ++Y)
```

```
plus_B(2,1)
        =  si 1 = 0 alors 2 sinon --plus_B(2, ++1)
        =   --plus_B(2, ++1)
        =   --plus_B(2, 2)
        =   --(si 2 = 0 alors 2 sinon --plus_B(2, ++2))
        =   -- --plus_B(2, ++2)
        =   -- --plus_B(2, 3)
        =  ...(BOUCLE !)
```

machine de Peano évoluée

Expressions

(i) 0, 1, 2, 3, 4, ...
(ii) ++ Expr
(iii) −− Expr
(iv) si $E_1 = E_2$ alors E_3 sinon E_4
(v) appel de fonction
(vi) $E_1 + E_2$

machine de Peano : exemples de définitions

Multiplication : algorithme

```
X * 0  =  0
X * 1  =  X
X * Y  =  (X * (Y - 1)) + X
```

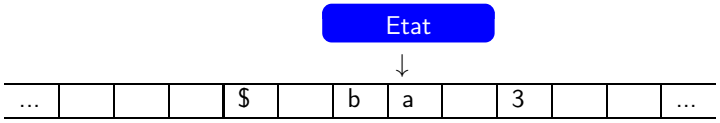
Multiplication : programme

```
mult(X,Y) = si Y = 0 alors 0 sinon
              si Y = 1 alors X sinon
                plus(X, mult(X,--Y))
```

machine de Peano encore plus évoluée...

Machine de Turing

- un ruban infini, chaque case du ruban peut contenir un symbole (lettre, chiffre, ...) ou être vide ;
- une tête de lecture/écriture qui pointe sur une case du ruban ;
- la tête de lecture/écriture est dans un état parmi une liste d'états possibles.
- Variantes : machine à plusieurs rubans, rubans semi-infinis, ...



Fonctionnement d'une machine de Turing

- Exécute les instructions d'un programme tant que cela est possible.

Programme d'une machine de Turing

Programme d'une machine de Turing : liste d'instructions

Les instructions sont du type :

si l'état est E et le symbole sous la tête est s alors

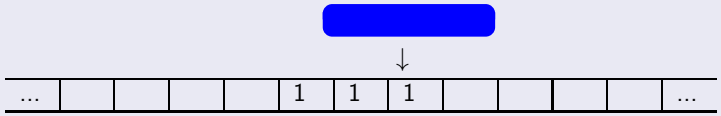
- remplacer s par s' ;
- passer dans l'état E'
- et déplacer la tête.

- Déplacement de la tête de lecture/écriture sur le ruban : une case à droite (noté $\rightarrow$) ou une case à gauche (noté $\leftarrow$).
- Notation d'une instruction (par ex) : $(E, s) \Longrightarrow (E', s', \rightarrow)$.

Exemple 1

Problème

Calculer $n+1$; le ruban contient initialement n écrit en base 1. La tête de lecture est sur le '1' le plus à droite.



Nombres en base 1

base 10	1	2	3	4	...	6	...
base 1	1	11	111	1111	...	111111	...

Algorithme possible

Se déplacer à gauche jusqu'à trouver une case vide et y mettre un '1'.

Programme de la machine de Turing correspondante

États

- Déplacement
- Stop

Instructions

1. (Déplacement , 1) $\Longrightarrow$ (Déplacement , 1, $\leftarrow$) : Si l'état est Déplacement et que la tête de lecture est sur '1' alors se déplacer d'une case à gauche.
2. (Déplacement , $\emptyset$) $\Longrightarrow$ (Stop , 1, $\leftarrow$) : Si l'état est Déplacement et que la tête de lecture est sur une case vide alors écrire '1' et passer dans l'état Stop .

Exemple 1

Problème

Calculer $n+1$; le ruban contient initialement n écrit en base 1. La tête de lecture est sur le '1' le plus à droite.

Instructions

1. (Déplacement , 1) $\Longrightarrow$ (Déplacement , 1, $\leftarrow$)
2. (Déplacement , $\emptyset$) $\Longrightarrow$ (Stop , 1, $\leftarrow$)

Exemple 2

Problème

Calculer $n+1$; le ruban contient initialement n écrit en base 10. La tête de lecture est sur le chiffre le plus à droite.

Exemples

1. $34+1$
2. $39+1$
3. $99+1$

Solution

États

- Ajoute : il faut ajouter 1 à la case courante
- Stop : c'est fini

Instructions

1. $(\text{Ajoute}, i) \Longrightarrow (\text{Stop}, i+1, \leftarrow)$ pour $i \in \{0,1,2,3,4,5,6,7,8\}$;
2. $(\text{Ajoute}, 9) \Longrightarrow (\text{Ajoute}, 0, \leftarrow)$.
3. $(\text{Ajoute}, \emptyset) \Longrightarrow (\text{Stop}, 1, \leftarrow)$

Solution

États

- Ajoute : il faut ajouter 1 à la case courante
- Stop : c'est fini

Instructions

1. $(\text{Ajoute}, 0) \Longrightarrow (\text{Stop}, 1, \leftarrow)$
   $(\text{Ajoute}, 1) \Longrightarrow (\text{Stop}, 2, \leftarrow)$
   $(\text{Ajoute}, 2) \Longrightarrow (\text{Stop}, 3, \leftarrow)$
   ...
   $(\text{Ajoute}, 7) \Longrightarrow (\text{Stop}, 8, \leftarrow)$
   $(\text{Ajoute}, 8) \Longrightarrow (\text{Stop}, 9, \leftarrow)$
2. $(\text{Ajoute}, 9) \Longrightarrow (\text{Ajoute}, 0, \leftarrow)$.
3. $(\text{Ajoute}, \emptyset) \Longrightarrow (\text{Stop}, 1, \leftarrow)$

Une machine un peu particulière

Instructions

1. $(\text{Ajoute}, i) \Longrightarrow (\text{Stop}, i+1, \leftarrow)$ pour $i \in \{0,1,2,3,4,5,6,7,8\}$;
2. $(\text{Ajoute}, 9) \Longrightarrow (\text{Ajoute}, 0, \leftarrow)$.
3. $(\text{Ajoute}, \emptyset) \Longrightarrow (\text{Ajoute}, 1, \leftarrow)$

Cette machine ne s'arrête jamais dans certains cas... Lesquels ?

En résumé

- Une machine de Turing prend en entrée une suite de symboles (sur le ruban).
- Soit la machine s'arrête, soit elle ne s'arrête pas.
- Si elle s'arrête, résultat sur le ruban : suite de symboles.
- Exemples :
  - Entrée : n en base 1, sortie : $n+1$ en base 1.
  - Entrée : n en base 10, sortie : $n+1$ en base 10.
  - Entrée : 2 entiers n et m, sortie : $n+m$.
  - Entrée : formule, par ex $(45+37)*12$, sortie : résultat.
  - ...

Calculabilité : nul n'est jamais assez fort pour ce calcul*

* : Luc Étienne, *L'Art du contrepet*

Une machine de Turing calcule une fonction

- Une suite de symbole $\Longleftrightarrow$ un entier.
- Donc machine de Turing :
  - Entrée : un entier.
  - Sortie : un entier.
- Autrement dit, une machine de Turing calcule une fonction de $\mathbb{N}$ dans $\mathbb{N}$ (si elle s'arrête).
  - Une machine de Turing $\Longleftrightarrow$ une fonction f.
  - Entrée de la machine : n, sortie : $f(n)$.

Thèse de Church-Turing

- Pour tout algorithme (fonction intuitivement calculable), il existe une machine de Turing qui l'implémente !
- Pour chacune des machines réelles, il existe une machine de Turing qui résout le même problème.

Fonctions calculables

Une fonction f est calculable si il existe une machine de Turing qui calcule f, i.e. :

- Si le ruban de la machine contient initialement un entier n quelconque alors :
  - La machine s'arrête ;
  - le ruban contient à la fin $f(n)$.

Fonctions calculables

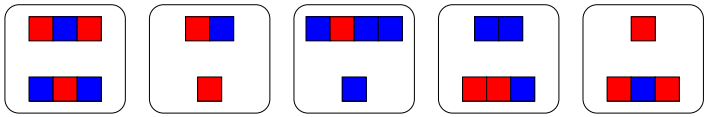
Toutes les fonctions de $\mathbb{N}$ dans $\mathbb{N}$ sont elles calculables ?

- Il y a autant de machines de Turing que de nombres entiers.
- Il y a autant de fonctions de $\mathbb{N}$ dans $\mathbb{N}$ que de nombres réels.
- Il y a strictement plus de réels que d'entiers...

Donc il y a strictement plus de fonctions de $\mathbb{N}$ dans $\mathbb{N}$ que de machines de Turing, **donc il existe des fonctions non calculables.**

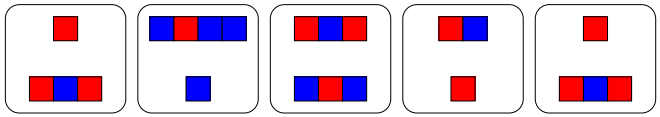
Un problème non calculable

Problème de correspondance de Post : dominos à aligner

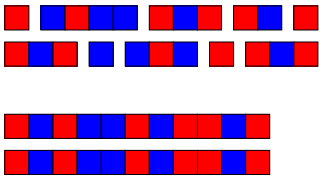


Question : existe-t-il un alignement des dominos tel qu'il y ait la même chose en haut et en bas ? (les répétitions de dominos sont autorisées)

Une solution : 5,3,1,2,5



Plus clairement :



D'autres problèmes non calculables

1. Ce programme s'arrête-t-il ?
2. Ce programme va-t-il me demander mon mot de passe ?
3. Si j'exécute ce programme, va-t-il effacer mon disque dur ?
4. Ce programme est-il un virus ?

En fait, la plupart des questions intéressantes sur les programmes sont non calculables !

Une lueur d'espoir...

▷ *Non calculable* ne veut pas dire *impossible à calculer* : existence de cas particuliers.
▷ D'autres problèmes au moins aussi intéressants sont calculables...

Conception d'algorithmes (et de programmes)

Mission

Concevoir un nouvel algorithme de multiplication de deux entiers utilisant **seulement**

1. la multiplication par 2,
2. la division par 2,
3. l'addition d'entiers,
4. le test de parité.

Un peu de réflexion...

Arithmétique...

Écriture binaire d'un entier n : avec des 0 et des 1 uniquement

$$n = n_k.2^k + \cdots + n_1.2^1 + n_0.2^0 = \sum_{i=0}^{k} n_i.2^i$$

(indice i : **poids du chiffre**)

Exemples

$13 = 8 + 4 + 1 = 1.2^3 + 1.2^2 + 0.2^1 + 1.2^0 (= 1101_2)$
$21 = 16 + 4 + 1 = 1.2^4 + 0.2^3 + 1.2^2 + 0.2^1 + 1.2^0 (= 10101_2)$
$10 = 8 + 2 = 1.2^3 + 0.2^2 + 1.2^1 + 0.2^0 (= 1010_2)$

Un peu plus de réflexion...

Soient X et Y deux entiers à multiplier. On a $Y = \sum_{i=0}^{n} y_i 2^i$

Du coup

$$\begin{aligned} X \times Y &= X \times \sum_{i=0}^{n} y_i 2^i \\ &= X \times (y_n.2^n + \cdots + y_1.2^1 + y_0.2^0) \\ &= X.y_n.2^n + \cdots + X.y_1.2^1 + X.y_0.2^0 \\ &= \sum_{i=0}^{n} y_i.X.2^i \end{aligned}$$

Exemple

$X = 35, Y = 26$
$Y = 16 + 8 + 2 = 1.2^4 + 1.2^3 + 0.2^2 + 1.2^1 + 0.2^0 \ (= \ 11010_2)$

i	y_i	2^i	$X.2^i$
0	0	1	35
1	1	2	70
2	0	4	140
3	1	8	280
4	1	16	560

Pour obtenir 35×26, il suffit d'additionner les lignes où $y_i \neq 0$:

$$35 \times 26 = 70 + 280 + 560 = 910$$

Encore plus de réflexion...

Comment savoir si y_i est nul ou pas ?

En base 10

- si n est un multiple de 10, il se termine par 0
- si $\frac{n}{10}$ est un multiple de 10, il se termine par 0 : l'avant-dernier chiffre de n est 0
- si $\frac{n}{10^k}$ est un multiple de 10, il se termine par 0 : le chiffre de poids k de n est 0
- exemples : 5021, 2301, 602215, ...

Encore plus de réflexion. . .

Comment savoir si y_i est nul ou pas ?

En base 2

- si n est pair, son écriture binaire se termine par 0
- si $\frac{n}{2^k}$ est pair, son écriture binaire se termine par 0 : le chiffre de poids k de n est 0
- si $y_i \neq 0$, c'est que $y_i = 1$

Il suffit de diviser Y par 2, jusqu'à arriver à 1 :

i	y_i	2^i	$Y/2^i$
0	0	1	26
1	1	2	13
2	0	4	6
3	1	8	3
4	1	16	1

Algorithme de calcul d'un produit

On calcule dans Z le produit de X et Y :

(a) mettre Z à zéro ;
(b) si $Y = 0$, c'est fini, Z contient le résultat ;
(c) si Y est impair, ajouter X à Z ;
(d) multiplier X par 2 ;
(e) diviser Y par 2 ;
(f) reprendre en (b).

Exemple et disposition pratique

$X = 61,\ Y = 37$

X	Y	Z
61	37	0
		61
122	18	
		61
244	9	
		305
488	4	
		305
976	2	
		305
1952	1	
		2257
3904	0	

Faisabilité pratique : cryptographie

Codage d'un texte : Jules César

Principe : décalage des lettres (code *monoalphabétique*)

ABCDEFGHIJKLMNOPQRSTUVWXYZ
GHIJKLMNOPQRSTUVWXYZABCDEF

Exemple : ROT13

ABCDEFGHIJKLMNOPQRSTUVWXYZ
NOPQRSTUVWXYZABCDEFGHIJKLM

Faiblesse : seulement 26 possibilités...

Codage d'un texte : Blaise de Vigenère

Principe : une clé donne le décalage (code *polyalphabétique*)
Exemple : la clé est INFO

CE COURS EST INTERESSANT
IN FOINF OIN FOINFOINFOI
LS IDDFY TBH OCCSXTBGGCC

Texte codé final :

LSI DDF YTB HOC CSX TBG GCC

Vigenère : avantages et inconvénients

Force : incassable si clé et texte de même longueur (**one-time pad**)

$$\text{OUI} + \text{ABC} = \text{PWL} = \text{NON} + \text{BHX}$$

Faiblesse :

- facilement (!) cryptanalysé pour les clés courtes utilisées en pratique,
- problème de gestion des clés : système privé, les utilisateurs doivent s'entendre sur les clés.

Cryptographie à clés publiques : principe

Principe

- chaque utilisateur a 2 clés : une publique P, une privée p ;
- un message codé avec P doit être décodé avec p ;
- un message codé avec p doit être décodé avec P ;
- P difficile à obtenir à partir de p (et inversement).

Avantage

Chaque utilisateur doit connaître

- sa clé privée,
- les clés publiques des autres utilisateurs (annuaire).

Cryptographie à clés publiques : utilisation

Envoi d'un message

- si Alice veut envoyer M à Bob, elle le code avec la clé publique de Bob ;
- lorsque Bob reçoit le message, il le décode avec sa clé privée.

Signature électronique

- Alice veut envoyer le message M, **signé**, à Bob ;
- Alice code M avec sa clé privée, elle obtient S ;
- elle envoie (M, S) à Bob ;
- Bob décode S avec la clé publique d'Alice ;
- s'il obtient M tout va bien, il est sûr que l'expéditeur est Alice ;
- sinon erreur de transmission, ou masquarade

Cryptographie à clés publiques : RSA

- Rivest, Shamir et Adleman, 1977
- un peu d'arithmétique : on peut trouver des entiers a, b, N tels que, pour tout entier x,

$$(x^a)^b = (x^b)^a = x \qquad (N)$$

  Par exemple, $N = 91,\ a = 5,\ b = 29$
- clé publique (a, N), clé privée (b, N)
- pour retrouver a à partir de b et N, il faut d'abord factoriser N ;
- on sait que $N = p_1 \times p_2$, p_1 et p_2 premiers ;
- **sécurité :**
  - en 2005, factorisation d'un entier de 663 bits ;
  - sur un PC, factorisation de 256 bits en un jour ;
  - clés courantes de RSA : 1024 ou 2048 bits...

Problème pour RSA

On souhaite calculer x^n, pour de **grandes** valeurs de n ($n > 10^{500}$)

Première approche

$$\begin{cases} x^1 & = & x \\ x^{n+1} & = & x \times x^n \end{cases}$$

ou bien, équivalemment,

$$x^n = \prod_{i=1}^{n} x$$

Ceci se fait en $n - 1$ multiplications

Banc d'essai :

- on utilise SEQUOIA, 10^{16} multiplications par seconde
- $n = 10^{50}$ (minimum pour RSA)
- pas de résultats au bout de 3 heures...
- pas de résultats au bout de 3 jours...
- pas de résultats au bout de 3 semaines...

Une première réaction

- *La machine est trop lente, il en faut une plus rapide*
- Combien de temps doit prendre une multiplication pour que le calcul de $x^{10^{50}}$ s'effectue en un jour (soit 10^5 secondes) ?
- On doit avoir

$$t_{\text{mult}} \times 10^{50} = 10^5 \text{ secondes}$$

c'est-à-dire

$$t_{\text{mult}} = \frac{10^5}{10^{50}} = 10^{-45} \text{ seconde}$$

Estimation du temps de calcul

1. pour calculer $x^{10^{50}}$, la méthode effectue $10^{50} - 1$ multiplications ;
2. SEQUOIA effectue 10^{16} multiplications par seconde ;
3. il faut donc
$$\frac{10^{50}}{10^{16}} = 10^{34} \text{ secondes}$$
pour ce calcul ;
4. on sait que 1 an $\approx 3.10^7$ secondes
5. il faut donc
$$\frac{10^{34}}{3.10^7} \approx 3.10^{26} \text{ ans}$$
pour ce calcul.

Remarque : dans la vraie vie, n est plutôt proche de 10^{500}...

- **Loi de Moore :** *La puissance des machines double tous les 18 mois* ;
- Une génération de machines dure donc 18 mois ;
- Au bout de combien de générations de machines peut-on espérer faire une multiplication en 10^{-45} seconde ?
- À la k-ième génération, on fait $10^{16} \times 2^k$ multiplications par seconde et une multiplication prend donc $\frac{10^{-16}}{2^k}$ seconde ;

- Une multiplication se fera donc en 10^{-45} seconde à la génération k tel que

$$\frac{10^{-16}}{2^k} = 10^{-45}$$

- c'est-à-dire

$$2^k = 10^{29}$$

- c'est-à-dire encore

$$k = \log_2 10^{29} \approx 97$$

- Une génération dure 18 mois $= 1,5$ ans ;
- Cette génération de machines verrait le jour dans $97 \times 1,5 = 145,5$ ans, **si aucune limite physique n'est rencontrée d'ici là**.

$\Longrightarrow$ on ne peut pas utiliser RSA ? ? ?

Plus malin. . .

$$\begin{cases} x^0 &= 1 \\ x^1 &= x \\ x^{2n} &= x^n \times x^n \\ x^{2n+1} &= x^n \times x^n \times x \end{cases}$$

Ceci se fait en (au plus) (environ) $2\lfloor \log_2 n \rfloor$ multiplications

Temps de calcul de $x^{10^{50}}$

- au plus environ $2\lfloor \log_2 10^{50} \rfloor$ multiplications ;
- $\lfloor \log_2 10^{50} \rfloor = 166$;
- donc au plus 333 multiplications ;
- si 10^9 multiplications par seconde, calcul effectué en $0,333$ μs. . .

Exemple plus réaliste

- $n = 10^{500}$
- 10^3 multiplications par seconde ;
- au plus $3,321$ secondes.

Moralité

▷ tout n'est pas calculable,
▷ et même si c'est calculable, c'est peut-être inaccessible :

$$10!\ \mu s \approx 3,6\ s$$

$$20!\ \mu s \approx$$

▷ un peu de réflexion ne peut pas nuire. . .